

# Disk storage management for LHCb based on Data Popularity estimator

**Mikhail Hushchyn**

Yandex School of Data Analysis, Moscow, Russian Federation  
Moscow Institute of Physics and Technology, Moscow, Russian Federation  
E-mail: [mikhail91@yandex-team.ru](mailto:mikhail91@yandex-team.ru)

**Philippe Charpentier**

CERN - European Organization for Nuclear Research  
E-mail: [Philippe.Charpentier@cern.ch](mailto:Philippe.Charpentier@cern.ch)

**Andrey Ustyuzhanin**

Yandex School of Data Analysis, Moscow, Russian Federation  
National Research University Higher School of Economics (HSE), Moscow, Russian Federation  
NRC "Kurchatov Institute", Moscow, Russian Federation  
Moscow Institute of Physics and Technology, Moscow, Russian Federation  
E-mail: [anaderi@yandex-team.ru](mailto:anaderi@yandex-team.ru)

**Abstract.** This paper presents an algorithm providing recommendations for optimizing the LHCb data storage. The LHCb data storage system is a hybrid system. All datasets are kept as archives on magnetic tapes. The most popular datasets are kept on disks. The algorithm takes the dataset usage history and metadata (size, type, configuration etc.) to generate a recommendation report. This article presents how we use machine learning algorithms to predict future data popularity. Using these predictions it is possible to estimate which datasets should be removed from disk. We use regression algorithms and time series analysis to find the optimal number of replicas for datasets that are kept on disk. Based on the data popularity and the number of replicas optimization, the algorithm minimizes a loss function to find the optimal data distribution. The loss function represents all requirements for data distribution in the data storage system. We demonstrate how our algorithm helps to save disk space and to reduce waiting times for jobs using this data.

## 1. Introduction

The LHCb collaboration is one of the four major experiments at the Large Hadron Collider at CERN. The detector, as well as the Monte Carlo simulations of physics events, create vast amount of data every year. This data is kept on disk and tape storage systems. Disks are used for storing data used by physicists for analysis. They are much faster than tapes, but are way more expensive and hence disk space is limited. Therefore it is highly important to identify which datasets should be kept on disk and which ones should only be kept as archives on tape. Currently, the data volumes on disk and tape are about 10.5 PB and 1.5 PB respectively. The

algorithm presented here is designed to select the datasets which may be used in the future and thus should remain on disk. Input information to the algorithm are the dataset usage history and dataset metadata (size, type, configuration etc.).

The algorithm consists of three separate modules. The first one is the Data Popularity Estimator. This module predicts the dataset future popularity by applying a machine learning algorithm to the algorithm's input information. The data popularity represents the probability for a dataset to be useful in future. Based on data popularity it is possible to identify which datasets can be removed from disk.

The second module is the Data Intensity Predictor. This module is needed to predict the future usage intensity of each dataset. Time series analysis and regression algorithms are used to make these predictions. Input information for this module is the dataset usage history.

The third module is the Data Placement Optimizer. In this module the data popularity and the predicted future usage intensities are used to estimate which datasets should be kept on disk and how many replicas they should have. For this purpose a loss function minimization problem is solved. The loss function represents all requirements for data distribution in the data storage system.

These three modules are described in detail in the following sections. In the results section we then show a comparison of our algorithm with a simple Last Recently Used (LRU) algorithm.

## 2. Related works

A Data Management Algorithm for hybrid hard disk drive (HDD) + solid-state drive (SSD) data storage system is described in [2]. The authors presents a method that shuffles datasets across storage tiers to optimize the data access performance. The method uses Markov chains[1] to predict the popularity of dataset accesses. The dataset placement optimization problem is solved based on the dataset accesses popularity.

A Popularity-Based Prediction and Data Redistribution Tool for the ATLAS Distributed Data Management is presented in [3,4]. The authors use artificial neural networks (ann)[1] to predict possible dataset accesses in the near-term future based on the dataset usage history. Then these predictions are used to redistribute data on the grid, i.e., adding and removing replicas.

A feature of our study is that dataset usage history in LHCb has a rather low statistics. The Data Management Algorithm from [2] needs more statistics for a good performance. The artificial neural networks from articles [3,4] are too complicated for our data and as a consequence an overfitting problem[1] may occur.

## 3. Input information

Dataset usage history and metadata are used as input information to the algorithm. In this study we use weekly dataset usage counters collected over the last two years. Dataset usage history represents as time series of 104 points. Each point represents the number of dataset usages during one week (i.e. the number of files accessed by Grid jobs divided by the number of files in the dataset).

The dataset metadata contains additional dataset information likes: the origin, the detector configuration, the file type, the data type (Monte Carlo simulations or real data), the event type, the creation week, the first usage week, the last usage week, the size for one replica, the total size of occupied disk space, the number of replicas on disk and some others.

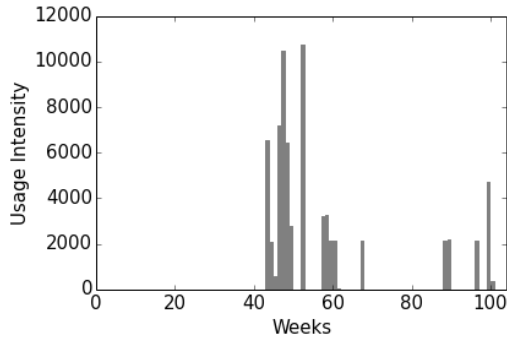
The algorithm takes as input a file which contains the dataset usage history and the dataset metadata. This file comes from the file catalogue.

## 4. Data Popularity Estimator

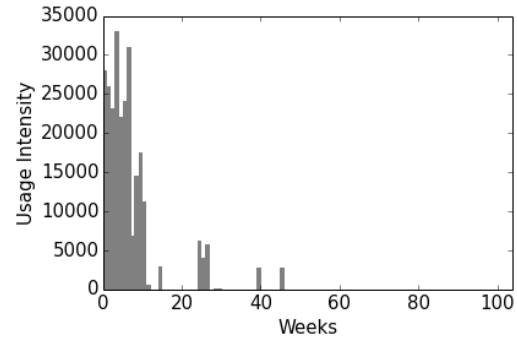
The Data Popularity Estimator module uses a classifier to calculate the data popularity. The classifier is a supervised machine learning algorithm[1] and consists of several steps. The following subsections describe each step of data popularity estimation.

### 4.1. Dataset labels

As the classifier is a supervised machine learning algorithm, each dataset should be labelled as popular or unpopular. The time series of dataset usage history are very sparse, therefore the last 26 weeks of usage history are used to label the data. If a dataset has not been used during the last 26 weeks we label it as unpopular and assign it a label value "1". Otherwise, the dataset is labelled as popular with a label value "0". This label defines the class of the dataset (0 for popular, 1 for unpopular). The figures 1 and 2 show each the time series of one dataset of each class.



**Figure 1.** Time series with label "0".



**Figure 2.** Time series with label "1".

### 4.2. Data preprocessing

The dataset metadata are used as input parameters for a classifier. Some new parameters are computed and used in the analysis as well as the existing ones. These factors describe the shape of the time series of the dataset usage history. While the last 26 weeks of the time series are used to label the datasets, the first 78 weeks are used to compute these new parameters. These parameters are *nb\_peaks*, *last\_zeros*, *inter\_max*, *inter\_mean*, *inter\_std*, *inter\_rel*, *mass\_center*, *mass\_center\_sqrt*, *mass\_moment* and *r\_moment*.

*Nb\_peaks* is the number of weeks during which a dataset has been used. *Last\_zeros* is the number of weeks since when the dataset was last used. *Inter\_max*, *inter\_mean*, *inter\_std* are the maximum value, the mean value and the standard deviation of the number of weeks between consecutive weeks of usage. *Inter\_rel* is the ratio of the *inter\_std* and *inter\_mean* values. *Mass\_center* is the center of gravity of a time series for a dataset, where the "mass" is the number of accesses to the dataset for each week. *Mass\_center\_sqrt*, *mass\_moment* and *r\_moment* are similar to *mass\_center*, but "mass" and "coordinate" have different degrees.

These parameters significantly increase the classifier's quality.

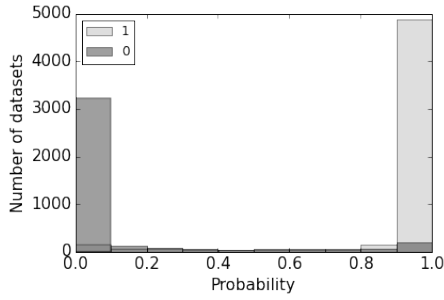
### 4.3. The classifier training

The new parameters, the dataset metadata and their labels are used to train a *Gradient Boosting Classifier*[1]. All datasets are split into two equal halves, i.e. half the datasets goes to the first half, the other to the second one. The classifier is trained on one half of the datasets and then

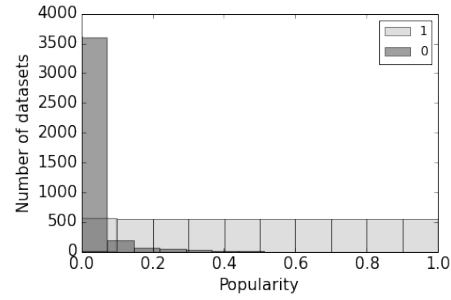
is used to predict probabilities to have label "1" for the second half of the datasets. The figure 3 shows the distribution of the probabilities for each class of datasets.

#### 4.4. Popularity estimation

The probability described previously is then transformed into a popularity estimator such that the popularity for datasets which have label "1" is uniform. The closer the popularity is to 1 the higher is the probability that it will be unused in the future. In this sense it is rather an 'unpopularity' estimator. The figure 4 represents the distribution of the popularity for each dataset class.



**Figure 3.** Distributions of the probability to have label "1" for each dataset class.



**Figure 4.** Distributions of the popularity for each dataset class.

## 5. Data Intensity Predictor

The data popularity represents the probability that a dataset will be unused in the future. Another important feature is predicted dataset usage intensity. There is a number of time series analysis algorithms that predict future values of time series. Since time series in this study have lack of statistics, parametric models such as polynomial regression, autoregression, ARMA and ARIMA models, artificial neuron networks (ANNs) and others are not suitable. This section shows how to use two non-parametric models to predict the dataset usage intensities. These models are Nadaraya-Watson kernel smoothing[1] and rolling mean values[1].

### 5.1. Nadaraya-Watson kernel smoothing

Let points  $(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)$  represent a time series and  $X^l = \{x_1, x_2, \dots, x_l\}$ . Then, the Nadaraya-Watson equation for kernel smoothing is:

$$\hat{y}_h(x; X^l) = \frac{\sum_{i=1}^l y_i K\left(\frac{\rho(x, x_i)}{h}\right)}{\sum_{i=1}^l K\left(\frac{\rho(x, x_i)}{h}\right)}, \quad (1)$$

where

$\hat{y}_h(x; X^l)$  is the time series value at  $x$  after kernel smoothing of  $X^l$  values,  
 $K\left(\frac{\rho(x, x_i)}{h}\right) = \exp\left(-\frac{(x-x_i)^2}{2h^2}\right)$  is the RBF smoothing kernel,  
 $h$  is the smoothing window width.

For the smoothing window width optimization the *Leave-One-Out*[1] method was applied:

$$LOO(h, X^l) = \sum_{i=1}^l (\hat{y}_h(x_i; X^l \setminus \{x_i\}) - y_i)^2 \mapsto \min_h \quad (2)$$

The *Nadaraya-Watson* equation for kernel smoothing with *LOO* smoothing window width optimization is applied to time series of dataset usage history. The maximum smoothing window width is 30 weeks. The figure 5 shows an example of time series after this smoothing is applied.

### 5.2. Rolling mean values calculation

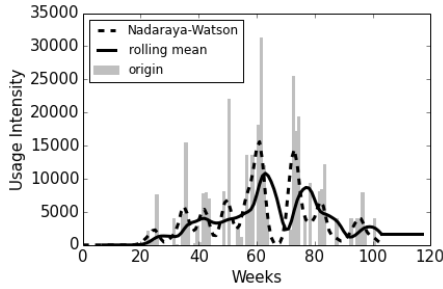
On the next step rolling mean values are calculated for additional smoothing of time series of dataset usage history. Let points  $(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)$  represent a time series after the kernel smoothing. Then, rolling mean values are defined as:

$$\hat{y}_k = \frac{\sum_{i=k-w}^k y_i}{w} \quad (3)$$

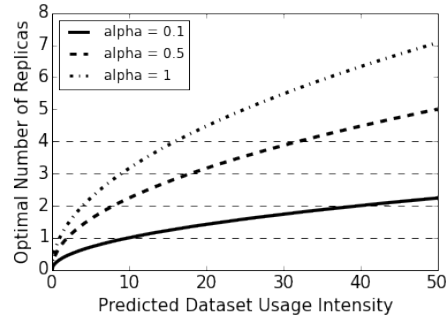
where  $w$  is the width of the moving window.

The window width is chosen such that 90% of all time series with equal *nb\_peaks* values have *inter\_max* values less than the window width.

The rolling mean value at moment  $x_i$  represents the dataset usage intensity at that moment. The simplest way to predict future dataset usage intensity is to take dataset usage intensity on last observation as future one. An example of calculated rolling mean values and predicted dataset usage intensity are shown on the figure 5.



**Figure 5.** Example of time series after *Nadaraya-Watson* kernel smoothing and rolling mean values calculation.



**Figure 6.** Dependence of optimal number of dataset replicas ( $Rp$ ) from its predicted usage intensity ( $I$ ) and  $\alpha$ .

## 6. Data Placement Optimizer

This section describes how one can estimate which dataset should be kept on disk and how many replicas they should have using the popularity and the predicted usage intensity for this dataset. Since disk space is more expensive than tapes, we would like to take a minimum of disk space. But on the other hand it is highly undesirable to remove from disk datasets which will be used in future. Additionally, we would like to create more replicas for the most popular datasets in order to reduce their average access time.

The requirements above are represented by the following loss function:

$$L = C_{disk} \sum_i^n S_i (Rp_i + \alpha \frac{I_i}{Rp_i}) \delta_i + C_{tape} \sum_i^n S_i (1 - \delta_i) + C_{miss} \sum_i^n S_i m_i, \quad (4)$$

$C_{disk}$  - cost of 1 Gb disk storage,

$C_{tape}$  - cost of 1 Gb tape storage,  
 $C_{miss}$  - cost of restoring 1 Gb data from tape to disk,  
 $\alpha$  - penalty for low number of replicas,  
 $S_i$  - size of one replica of  $i^{th}$  dataset,  
 $Rp_i$  - number of replicas of  $i^{th}$  dataset,  
 $I_i$  - predicted usage intensity of  $i^{th}$  dataset;  
 $\delta_i$  is equal to 1 if  $i^{th}$  dataset is on disk, otherwise it is 0;  
 $m_i$  is equal to 1 if  $i^{th}$  dataset was restored from tape to disk.

The first term of the loss function represents the cost of storage of the datasets on disk. The second term is the cost of storage of the datasets on tape. The last term is the cost of mistakes, when a dataset was removed from disk but then is used.

The expression in brackets in the first term of the loss function is used to find the optimal number of replicas for datasets on disk based on predicted usage intensities. The optimal number of replicas for a dataset with predicted usage intensity of  $I_i$  and for the value  $\alpha$  is

$$Rp_{i\_optimal} = \sqrt{\alpha I_i}, \quad (5)$$

The figure 6 shows how the optimal number of replicas for a dataset depends on its predicted usage intensity and alpha value. For example, suppose the predicted usage intensity for a dataset is  $I = 10$  usages per week and  $\alpha = 0.5$ . Then  $Rp_{optimal} = \sqrt{\alpha I} = \sqrt{0.5 * 10} = 2.24 \approx 2$  replicas.

The  $\delta_i$  value in the loss function depends on the data popularity threshold value. Datasets with popularities equal to or higher than this threshold value are removed from disk ( $\delta_i = 0$ ). The  $m_i$  value is the product of the  $1 - \delta_i$  and the label of  $i^{th}$  dataset (0 or 1).

The loss function optimization consists in finding the data popularity threshold value and dataset optimum number of replicas that provide the minimum value of the loss function.

## 7. Results

### 7.1. LRU algorithm

In this article we compare our algorithm with the Last Recently Used (LRU) algorithm. The LRU algorithm takes the last observations of the dataset usage history and decides which dataset should be removed from disk. In this study the first 78 weeks usage history time series are used as the algorithm inputs. The last 26 weeks are used to measure the quality of the algorithm. Thus if a data set was not used during the last  $N$  weeks (from  $78 - N^{th}$  to  $78^{th}$  weeks), this dataset is removed from disk. The number of disk replicas are not changed compared to the original number of replicas.

### 7.2. Downloading time

The following function is used to estimate the time of downloading of all datasets by all users (the generic term 'downloading' is used to represent an access to the dataset from a job):

$$T = \sum_{i=1}^n I_i^* S_i t_{disk} \alpha(Rp_i) \delta_i + \sum_{i=1}^n (K_{tape} + S_i t_{tape}) m_i + \sum_{i=1}^n I_i^* S_i t_{disk} m_i \quad (6)$$

where  $\alpha(Rp_i) = 0.05 + \frac{1}{Rp_i}$

$t_{disk}$  - average time of downloading 1 Gb of data from disk,

$t_{tape}$  - average time of downloading 1 Gb of data from tape to disk,

$K_{tape}$  - constant time needed to restore a dataset from tape to disk,

$I_i^*$  - average number of downloading of a dataset per week,

$S_i$  - size of one replica of  $i^{th}$  dataset,

$Rp_i$  - number of replicas of  $i^{th}$  dataset,  
 $\delta_i$  is equal 1 if the  $i^{th}$  dataset is on disk, otherwise it is 0,  
 $m_i$  (misclassification) is equal 1 if  $i^{th}$  dataset has to be restored from tape to disk.

The first term of the downloading time equation represents the time of download of all datasets from disk by all users. The second term represents the time needed to restore from tape datasets that were removed from disk due to an algorithms bad decision. The third term represents the time of download of restored datasets by all users. The first 78 weeks of the dataset usage history time series are used as algorithms inputs. The last 26 weeks are used to measure the quality of the algorithms and to estimate how many times the datasets were downloaded.

### 7.3. Algorithms comparison

Datasets which were created and first used earlier than 78<sup>th</sup> week are used to compare algorithms. The total number of datasets used for the comparison is 7375. In this paper we use rather pessimistic values of the parameters to emphasize that the disk space is highly limited. The following values of the parameters are used to optimize the loss function:  $C_{disk} = 100$ ,  $C_{tape} = 1$ ,  $C_{miss} = 2000$ . The values of the parameters for the downloading time function are  $t_{disk} = 0.1$  hour/Gb,  $t_{tape} = 3$  hours/Gb and  $K_{tape} = 24$  hours.  $C_{disk} \gg C_{tape}$  represents an idea that the disk space is limited.  $C_{miss} \gg C_{disk}$  means the number of restored datasets should be minimal.  $t_{tape} \gg t_{disk}$  and large  $K_{tape}$  value show that a dataset restoring from tape to disk takes a lot of time.

Tables 1 and 2 show results for our algorithm with 4 maximum dataset number of replicas and for the LRU algorithm. *Downloading time ratio* is the ratio of the downloading time after applying the algorithm to the original downloading time. *Saving space* column shows how much disk space can be saved using this algorithm. *Nb of wrong removings* column represents the number of datasets which are proposed to be removed from disk but are then used again in the future.

Both algorithms save about the same amount of disk space, but our algorithm has an extremely low number of mistakes. The tables show that our algorithm with 4 maximum dataset number of replicas slightly decreases the download time.

Table 3 demonstrates that for a maximum number of replicas of 7 our algorithm helps to save up to 40% of disk space and decreases the downloading time by up to 30%.

**Table 1.** Results for LRU algorithm.

N	Downloading time ratio	Saving space, %	Nb of wrong removings
1	1.33	63	1973
2	1.28	58	1659
5	1.4	50	1357
10	1.11	44	966
15	1.07	38	635
20	1.03	33	370
25	1.02	30	193

**Table 2.** Results for our algorithm with 4 maximum numbers of replicas.

Alpha	Downloading time ratio	Saving space, %	Nb of wrong removings
0	3.35	71	9
0.01	0.99	46	9
0.05	0.96	34	9
0.1	0.96	30	9
0.5	0.96	23	9
1	0.96	19	9
2	0.96	16	9

**Table 3.** Results for our algorithm with 7 maximum numbers of replicas.

Alpha	Downloading time ratio	Saving space, %	Nb of wrong removings
0	3.35	71	8
0.001	1.03	57	8
0.005	0.72	40	8
0.01	0.68	34	8
0.05	0.63	11	8
0.1	0.62	1	8

A python module implementing our algorithm and its web service can be downloaded from [5]. Our study is performed by means of a Reproducible Experiment Platform[6] - environment for conducting data-driven research in a consistent and reproducible way.

## 8. Conclusion

In this paper, we presented a study of developing the algorithm for disk storage management. The method presented here demonstrates how the algorithms of machine learning, regression and time series analysis can be used in data management of the LHCb data storage system. The results shows that our algorithm helps to save a significant amount of disk space and reduce the average downloading time.

## References

- [1] Hastie T, Tibshirani R, Friedman J 2009 *The Elements of Statistical Learning* (Berlin: Springer)
- [2] Lipeng W, Zheng L, Qing C, Feiyi W, Sarp O, Bradley S 2014 30<sup>th</sup> *Symposium on Mass Storage Systems and Technologies (MSST): SSD-optimized workload placement with adaptive learning and classification in HPC environments* (California: IEEE)
- [3] Beermann T, Stewart A, Maettig P 2014 *The International Symposium on Grids and Clouds (ISGC) 2014: A Popularity-Based Prediction and Data Redistribution Tool for ATLAS Distributed Data Management* (PoS) p 4
- [4] Beermann T 2013 *Popularity Prediction Tool for ATLAS Distributed Data Management: J. of Phys.: Conf. Ser. 513 (2014) 042004* (IOP Publishing)
- [5] *Python module and web service* URL <https://github.com/yandexdataschool/DataPopularity>
- [6] *Reproducible Experiment Platform (REP)* URL <https://github.com/yandex/rep>